EF 903 178 214 US

A

PTO/SB/05 (4/98)
Approved for use through 09/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE

Please type a plus sign (+) inside this box →

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 C.F.R. § 1.53(b))

Attorney Docket No.	AI9-98264
First Inventor or Application Identifier	David L Thornburg et al
Title	Method and apparatus for applying [illegible]
Express Mail Label No.	EF903178214US

JC6111 U.S. PTO 12/03/98

APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

ADDRESS TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231

1. [ ] * Fee Transmittal Form *(e.g., PTO/SB/17)* *(Submit an original and a duplicate for fee processing)*
2. [ ] Specification [Total Pages 53] *(preferred arrangement set forth below)*
   - Descriptive title of the Invention
   - Cross References to Related Applications
   - Statement Regarding Fed sponsored R & D
   - Reference to Microfiche Appendix
   - Background of the Invention
   - Brief Summary of the Invention
   - Brief Description of the Drawings *(if filed)*
   - Detailed Description
   - Claim(s)
   - Abstract of the Disclosure
3. [ ] Drawing(s) *(35 U.S.C. 113)* [Total Sheets 5]
4. Oath or Declaration [Total Pages 6]
   - a. [ ] Newly executed (original or copy)
   - b. [ ] Copy from a prior application (37 C.F.R. § 1.63(d)) *(for continuation/divisional with Box 16 completed)*
     - i. [ ] DELETION OF INVENTOR(S) Signed statement attached deleting inventor(s) named in the prior application, see 37 C.F.R. §§ 1.63(d)(2) and 1.33(b).

NOTE FOR ITEMS 1 & 13: IN ORDER TO BE ENTITLED TO PAY SMALL ENTITY FEES, A SMALL ENTITY STATEMENT IS REQUIRED (37 C.F.R. § 1.27), EXCEPT IF ONE FILED IN A PRIOR APPLICATION IS RELIED UPON (37 C.F.R. § 1.28).

5. [ ] Microfiche Computer Program *(Appendix)*
6. Nucleotide and/or Amino Acid Sequence Submission *(if applicable, all necessary)*
   - a. [ ] Computer Readable Copy
   - b. [ ] Paper Copy (identical to computer copy)
   - c. [ ] Statement verifying identity of above copies

JC551 U.S. PTO 09/204973 12/03/98

ACCOMPANYING APPLICATION PARTS

7. [✓] Assignment Papers (cover sheet & document(s))
8. [ ] 37 C.F.R.§3.73(b) Statement *(when there is an assignee)* [ ] Power of Attorney
9. [ ] English Translation Document *(if applicable)*
10. [ ] Information Disclosure Statement (IDS)/PTO-1449 [ ] Copies of IDS Citations
11. [ ] Preliminary Amendment
12. [✓] Return Receipt Postcard (MPEP 503) *(Should be specifically itemized)*
13. [ ] * Small Entity Statement(s) *(PTO/SB/09-12)* [ ] Statement filed in prior application, Status still proper and desired
14. [ ] Certified Copy of Priority Document(s) *(if foreign priority is claimed)*
15. [ ] Other: .....

16. **If a CONTINUING APPLICATION,** *check appropriate box, and supply the requisite information below and in a preliminary amendment:*

[ ] Continuation [ ] Divisional [ ] Continuation-in-part (CIP) of prior application No: _____ / _____

Prior application information: Examiner _____ *Group / Art Unit:* _____

For CONTINUATION or DIVISIONAL APPS only: The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 4b, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

17. CORRESPONDENCE ADDRESS

[ ] *Customer Number or Bar Code Label* *(Insert Customer No. or Attach bar code label here)* or [✓] *Correspondence address below*

Name	Carsten, Yee and Cahoon, LLP
Address	13760 Noel Rd., Suite 300 P.O. Box 802334
City	Dallas, Texas 75380
Country	USA
Telephone	972-362-2001
Zip Code	
Fax	

Name (Print/Type)	David A Mims, JR	Registration No. (Attorney/Agent)	32,708
Signature	David A. Mims, Jr.	Date	12/3/98

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Box Patent Application, Washington, DC 20231.

EF903178214US

PTO/SB/17 (2/98)
Approved for use through 9/30/2000. OMB 0651-0032
Patent and Trademark Office: U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

FEE TRANSMITTAL

Patent fees are subject to annual revision on October 1.
These are the fees effective October 1, 1997.
Small Entity payments must be supported by a small entity statement, otherwise large entity fees must be paid. See Forms PTO/SB/09-12.
See 37 C.F.R. §§ 1.27 and 1.28.

TOTAL AMOUNT OF PAYMENT ($) 3958.00

Complete if Known	
Application Number	
Filing Date	
First Named Inventor	David L. Ehnebuske et al.
Examiner Name	
Group / Art Unit	
Attorney Docket No.	AT9-98-266

METHOD OF PAYMENT (check one)

1. ☑ The Commissioner is hereby authorized to charge indicated fees and credit any over payments to:

Deposit Account Number: 09-0447
Deposit Account Name: IBM Corporation

☑ Charge Any Additional Fee Required Under 37 C.F.R. §§ 1.16 and 1.17
☐ Charge the Issue Fee Set in 37 C.F.R. § 1.18 at the Mailing of the Notice of Allowance

2. ☐ Payment Enclosed: ☐ Check ☐ Money Order ☐ Other

FEE CALCULATION

1. BASIC FILING FEE

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
101	790	201	395	Utility filing fee	760
106	330	206	165	Design filing fee	
107	540	207	270	Plant filing fee	
108	790	208	395	Reissue filing fee	
114	150	214	75	Provisional filing fee	

SUBTOTAL (1) ($) 760.

2. EXTRA CLAIM FEES

	Extra Claims		Fee from below	Fee Paid
Total Claims 98	-20** = 78	X	18	= 1404
Independent Claims 26	- 3** = 23	X	78	= 1794
Multiple Dependent				= -0-

**or number previously paid, if greater; For Reissues, see below

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description
103	22	203	11	Claims in excess of 20
102	82	202	41	Independent claims in excess of 3
104	270	204	135	Multiple dependent claim, if not paid
109	82	209	41	** Reissue independent claims over original patent
110	22	210	11	** Reissue claims in excess of 20 and over original patent

SUBTOTAL (2) ($) 3198.

FEE CALCULATION (continued)

3. ADDITIONAL FEES

Large Entity Fee Code	Large Entity Fee ($)	Small Entity Fee Code	Small Entity Fee ($)	Fee Description	Fee Paid
105	130	205	65	Surcharge - late filing fee or oath	
127	50	227	25	Surcharge - late provisional filing fee or cover sheet.	
139	130	139	130	Non-English specification	
147	2,520	147	2,520	For filing a request for reexamination	
112	920*	112	920*	Requesting publication of SIR prior to Examiner action	
113	1,840*	113	1,840*	Requesting publication of SIR after Examiner action	
115	110	215	55	Extension for reply within first month	
116	400	216	200	Extension for reply within second month	
117	950	217	475	Extension for reply within third month	
118	1,510	218	755	Extension for reply within fourth month	
128	2,060	228	1,030	Extension for reply within fifth month	
119	310	219	155	Notice of Appeal	
120	310	220	155	Filing a brief in support of an appeal	
121	270	221	135	Request for oral hearing	
138	1,510	138	1,510	Petition to institute a public use proceeding	
140	110	240	55	Petition to revive - unavoidable	
141	1,320	241	660	Petition to revive - unintentional	
142	1,320	242	660	Utility issue fee (or reissue)	
143	450	243	225	Design issue fee	
144	670	244	335	Plant issue fee	
122	130	122	130	Petitions to the Commissioner	
123	50	123	50	Petitions related to provisional applications	
126	240	126	240	Submission of Information Disclosure Stmt	
581	40	581	40	Recording each patent assignment per property (times number of properties)	
146	790	246	395	Filing a submission after final rejection (37 CFR 1.129(a))	
149	790	249	395	For each additional invention to be examined (37 CFR 1.129(b))	

Other fee (specify) _____

Other fee (specify) _____

*Reduced by Basic Filing Fee Paid

SUBTOTAL (3) ($)

SUBMITTED BY

				Complete (if applicable)	
Typed or Printed Name	David A. Mims, Jr.			Reg. Number	32,708
Signature	David A. Mims, Jr.	Date	12/3/98	Deposit Account User ID	

Burden Hour Statement: This form is estimated to take 0.2 hours to complete. Time will vary depending upon the needs of the individual case. Any comments on the amount of time you are required to complete this form should be sent to the Chief Information Officer, Patent and Trademark Office, Washington, DC 20231. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Assistant Commissioner for Patents, Washington, DC 20231.

Docket No. AT9-98-266

METHOD AND APPARATUS FOR APPLYING BUSINESS RULES IN AN OBJECT MODEL DRIVEN CONTEXT

CROSS REFERENCE TO RELATED APPLICATION

The present invention is related to applications entitled "Method And Apparatus For General Integrity Rule Checking Point In An Application," U. S. Patent Application No. _____ , Attorney Docket No. AT9-98-267, filed even date hereof, assigned to the same assignee; "System and Method And Data Processing System For Specifying And Applying Rules To Classification-Based Decision Points In An Application System," U. S. Patent Application No. _____, Attorney Docket No. AT9-98-287, filed even date hereof, assigned to the same assignee; and "Method and Apparatus for Identifying Applicable Business Rules," U. S. Patent Application No. 09/993,718, Filed 12/18/97, assigned to the same assignee and all of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

1. Technical Field:

The present invention relates to executing an enterprise application. More particularly, the present invention relates to executing an object-oriented enterprise application, which includes at least one method, which includes trigger or control points for attaching and running rules.

2. Description of Related Art:

Recently, businesses, particularly large enterprises, have moved toward object-oriented programming as a means to make the implementation of their business applications more flexible and adaptable to business environment and business practice changes. While this is a step forward compared to previous art, many businesses are finding that it is necessary to go beyond conventional object-oriented programming to achieve the flexibility and adaptability they require.

One approach to this is to externalize the highly variable business decisions into business rules, which are described and manipulated by business experts instead of developers. Applications entitled "Method And Apparatus For General Integrity Rule Checking Point In An Application," U. S. Patent Application No._____, Attorney Docket No. AT9-98-267, filed even date hereof, assigned to the same assignee; and "System and Method And Data Processing System For Specifying And Applying Rules To Classification-Based Decision Points In An Application System" U. S. Patent Application No. _____, Attorney Docket No. AT9-98-287, filed even date hereof, assigned to the same assignee, are two examples of this approach.

In designing and constructing an application, developers face an analogous problem. They, too, would like to be able to add to or modify the application's behavior without having to change the code of the application, but with a technical rather than business

intent. Two examples should suffice to demonstrate this. During the testing phases of development or when problems arise after an application has gone into production, it is often desirable to temporarily add functionality at specific points in the application's implementation object model. The functionality is added to check that particular technical invariants (the date and time of the last update of an object may not be earlier than an operation that it was before it) or constraints (a Person object may have no more than one spouse at any given point in time) imposed by the implementation object model are not being violated. The information can also be recorded internally to the application in a log for later analysis. Similarly, it often arises that it is desirable, particularly in "packaged" applications intended for use in multiple different enterprises, to be able to convert data between a form or forms that are convenient to the various end users and the form or forms used internally by the application.

The prior art cited above describes points of potential rule attachment (the "control points") in business terms, not terms related to the application's implementation object model. Nonetheless, a business rules facility can sometimes be used for developers' technical, as opposed to business, purposes; the added or altered functionality is implemented using the same mechanism used to implement externalized business rules. But doing so has three distinct disadvantages. First, adding technical rules to the business rules can be confusing for the business experts; not all of the rules

they see would then be business rules. This makes their work more difficult and error prone. Second, it requires that the application developers recast their technical problem in business terms to discover which, if any, existing business-oriented control points they can use to attach their technical rule. Third, the control points required to make the application flexible to business changes often do not necessarily occur where they need to for technical purposes. It is apparent, therefore, this approach is not adequately adapted for technical use by developers.

Another approach to this problem is to add the required new behavior to the system by using the well-known "Decorator," "Strategy," or "Template Method" patterns as taught by Erich Gamma, *et al*, "Design Patterns: Elements Of Reusable Object-Oriented Software By Gamma" *Addison-Wesley Publishing Company, ISBN 0-201063361-6, pp. 175-179.* Use of these techniques has the advantage for the developer of being tied directly to the application's implementation object model. For example, the Decorator pattern, in particular when used in a design that strictly separates object interfaces from object implementations, is often easy to use to cause a decorated object to exhibit arbitrary additional or modified behavior just before or just after any method of the decorated object. But all of these object-oriented coding patterns have the strong disadvantage that the added or altered functionality must be implemented by making programming changes, a more complex

and time consuming process than is changing an externally defined rule.

SUMMARY OF THE INVENTION

This invention provides a means for specifying, applying, and managing sets of temporary or permanent additions or modifications to the behavior of object-oriented programs without having to change the code of the program, by using externalized rules. The points at which the externalized rules may be applied is determined by the implementation object model, thus making their specification natural to the program developers who are familiar with the program's implementation object model.

This invention introduces the concept of dynamic method-based trigger or control points as a means for identifying potential rule attachment points in objects and identifying the rules that are applicable to each dynamic control point. A dynamic control point is a specialization of the general control point described in "Method and Apparatus for Identifying Applicable Business Rules," U.S. Application No. 09/993,718, filed 12/18/97, referenced above, adapted to provide rule attachment points.

Docket No. AT9-98-266

BRIEF DESCRIPTION OF THE DRAWINGS

The novel features believed characteristic of the invention are set forth in the appended claims. The invention itself, however, as well as a preferred mode of use, further objectives and advantages thereof, will best be understood by reference to the following detailed description of an illustrative embodiment when read in conjunction with the accompanying drawings, wherein:

Figure 1 illustrates the object method context relationship with the external rules;

Figure 2 illustrates an example of rule association to control point context;

Figure 3 illustrates a flow chart of the method in the present invention;

Figure 4 illustrates a flow chart example of the method in the present invention; and

Figure 5 illustrates the relationship between objects and their pre-method and post-method control points and to the rules associated with them.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Figure 1 illustrates a portion of a typical program's implementation object model. In **Figure 1**, two objects are present, **200** and **210**. As can be seen from object **200**, each object has some non-zero number of methods. In the present invention, each method has two method-type trigger or control points, a pre-method control point and a post-method control point. Each method is limited to exactly two of these control points. In objects of the present invention, each control point may have a plurality of external rules associated with it.

Therefore, in the present invention, an object may have a very large number of methods, the number of method-type control points is always the number of methods within the object times two.

Figure 1, illustrates software objects (**200** and **210**) containing a plurality of methods **201** to **203** and **211** to **213** and a set of external rules **220** to **223**.

Figure 2 further illustrates the relationship of the method control points to the external rule structure of the present invention. In the present invention, each object class has defined within it a number of methods. Each method has exactly two control points, a pre-method control point and a post-method control point. It can be seen from **Figure 2**, objects of class ENT have methods **1** -

Docket No. AT9-98-266

n, that each have a pre- and a post-method control point assigned to them.

Control points assigned to a particular method provide a nexus to the external rules. In **Figure 2**, they are exemplified by rules **A** to **C**. In the present invention, the developer associates a rule with a method or a context based on technical needs. For instance, method **1** has a pre- and a post-method control point. Looking now at rule **A**, the developer associates rule **A** with method **1**, method **2** and method **N**. Still describing rule **A**, note that each rule has been assigned to a specific control point on each method. For instance, rule **A** has been associated to **ENT** method **1**, pre-method control point (ENT.Method1.Pre.CP), **ENT** method **2** pre-method control point (ENT.Method2.Pre.CP), and **ENT** method **N** pre-method control point (ENT.MethodN.Pre.CP).

Pre- and post- merely designate that the control point comes before the method in time or after the method in time.

Note also in **Figure 2** that certain control points are flagged, designated in the figure by "flg" next to the flagged control points, while others remain un-flagged. Flagging denotes an active control point. Flagging is a means for a developer or the program itself to identify control points which run any rules associated with them. In certain instances it might be expedient to skip control points that have no rules or control points where the rules are unimportant or unneeded at the current time.

Figure 3 is a flowchart of the method in the present invention. **Figure 3**, as in all of the figures, is merely an example of the preferred embodiment of the present invention, and is not meant to limit the scope of the present invention. One of ordinary skill in the art would realize that other implementations of this method are possible.

In step **410** of **Figure 3**, an application is executed. The application referred to in step **410** is an object-oriented application, and in particular, is composed of objects. Each object is identified by class. In step **420**, a method associated with a particular object is invoked. Invoking a method requires knowing the class of the object and the name of the method being invoked. Just before the logic of the method is executed, the method's pre-method control point is encountered.

Next, in step **430**, the flag for this control point is checked to see whether the control point is active or not. If it is not, the method execution proceeds as if there were no control point present. Otherwise control passes to decision box **430**.

Decision box **430** is crucial to the present invention. As presently drawn in **Figure 3**, decision box **430** merely checks an activation flag to see if the control point is active. If it is active, the flow goes on to step **440** where the object and method names are used to select a rule. However, alternative embodiments are available with respect to decision block **430**.

Next, in step **440**, the object's class name, method name and the fact that this is a pre-method control

pointer are used to find applicable rules, if any. By knowing all possible rules associated with the object class and the method invoked on the object at step **450**, the rules for the control point can be selected. However, there may not be any rules associated with this particular control point. In fact, in most cases, rules are probably not assigned. In the present example, if there are no rules for the control point, the method will return to the method execution block **420** to continue the execution of the method as if there had been no control point encountered.

An important feature of the present invention is that different rules can be mapped to different control points. These rules are not exclusive to these control points, but can be used in conjunction with different kinds of control points.

As alluded to above, the types of rules associated with a single control point, of the sort that has been discussed, can perform a variety of different functions. In contrast to the present invention, prior art control points are intended to perform a specific predefined business decision. For this reason, the control points in the present invention are referred to as "method context control points" to distinguish them from the control points of prior art. An advantage of the present invention over prior art is that any method context control point can have a variety of different rules and different types of rules associated with it. The need for these rules can change over time. Although the skill level needed to adapt an object using method context

control points requires an understanding of the implementation object model and so is often beyond the skill possessed by business analysts, it requires substantially less skill than required to change the internals of objects.

In a preferred embodiment, a person involved in interpreting implementation object models can associate a plurality of different types of rules to a specific method context control point giving the enterprise the flexibility to invoke situation and time-dependent rules based on the implementation object model considerations rather than the business decision considerations supported by other types of control points.

In step **460**, rules associated with the method context control point are executed. Again, the rules associated with a single method context control point can perform a variety of functions. The functions themselves can change over time rather than being set and pre-defined as in prior art rules.

Another important feature of the present invention is that the function of the rules does not necessarily relate to the business enterprise itself. At any control point, a variety of types of rules, having a variety of functions, can be applied. Examples range from merely formatting or mechanical application type rules to rule types implementing exemplary objectives of the business. Any type of rule can be associated at control points and re-associated to other control points and associated to more than one control point.

Finally, in step **470**, a combining algorithm is used to combine the results of the rule implementation for each method context control point. This combining algorithm can be different for each method control point.

If the rules assess whether a series of constraints have been passed (which is a common case), each rule will return a boolean telling whether the constraint it checks has passed or failed. In these cases, it is common that the combining algorithm will be a logical 'and' so that the combined result is such that if all of the rules agree that the constraints are passed, the overall result is a 'pass', but if any of the rules determine the constraint it checks for has failed, the combined result is a 'fail'.

In step **480**, a determination is made whether the combined result of having run the rules is a decision that the rest of the method invoked in step **420** is to be processed or skipped. Commonly, this decision is taken simply by checking the boolean state of the combined result; if the result is a fail, the rest of the method is skipped. In such cases it is also common that, during this step, an exception is raised so that the normal error handling mechanisms of the application can detect that a constraint has been violated and take appropriate action - for example, telling the application's user or logging the fact of the constraint violation.

If, on the other hand, the decision is that the rest of the method is not to be bypassed, the method's execution is continued. For the common case of a combined result being a boolean stating that the

constraints are passed, the execution of the method continues and no result is communicated to the method. If, the combined result is of a different non-boolean type, the combined result is typically made available to the method execution so that it may integrate that value into its processing.

The processing shown in **Figure 3** shows the flow for a pre-method control point. As mentioned in the Summary of the Invention, the present invention also provides for post-method control points which occur just after the logic of the method completes. The flow for these is similar to that shown in **Figure 3** except that processing for post-method control points begins at the completion of the method execution instead of just before it starts.

Figure 4 illustrates a specific example of the implementation of the process of the present invention. While many possible business models are available, one might include a process for assigning a value of a vehicle for insurance purposes. In step **510**, the vehicle insurance application is invoked. In the business object model there is a vehicle business object. In assigning an estimated value for a vehicle, certain rules are necessary to validate the estimate. Examples of such rules include checking the estimate based on age, make, model, or the trust the organization has in the person doing the estimating. Which rules are found varies over time. In step **520**, assigning a value to a vehicle includes executing a 'set value' method. At this point a dollar amount for the vehicle value must be entered. Proper format for the value would be a dollar amount in

some number of digits, for instance, $10,000.00. In step **530**, the process determines if a control point associated with the vehicle.setValue context is active.

In step **530**, a determination is made whether this pre-method control point is active. In the current example, it is, and thus the flow proceeds to step **540**. Control points associated with rules other than testing rules are generally not deactivated, but may be for special purposes. To improve performance, control points which are known a prior to be associated with no rules may be deactivated.

Since the control point is active, the process moves into step **540** where rules associated with vehicle.setValue pre-method control point are selected. Once again these rules are chosen because they have been associated with this vehicle.setValue method context control point. As noted several times above, over time, different rules could be associated with the control points as needs change. In other words, using the method context control point process, a fixed point of variability is identified for placement of the rules without any consideration as to whether the rules will actually apply, or if they do apply, what their function might be.

In the preferred embodiment, each method has exactly two trigger points or control points, a pre-method and a post-method control point. Because the control points are regularly placed with respect to implementation object models - just before and just after each method - those familiar with an application's implementation

object model will immediately understand where, during application execution, rules may be placed to modify the behavior of the application. Further, they may do so without modifying the application's internal code.

Step **540** performs the same function as steps **440** and **450** of the preceding figure. In this step, the rules are found which are associated with the vehicle.setValue pre-method control point.

One of the important features of this invention is that the types of rules available for selection are very diverse. These types of rules not only include business rules, but also include functional and maintenance rules. In the present example, the rules are run (step **550**). Three types of example rules are given: a mechanical check which assesses the entry itself, a business type rule which determines if the entry value is sensible, and a third type of rule, which is completely unrelated to the other two rule types.

The method being executed in the present example saves the value of the vehicle. So, if the user inputs a vehicle value amount of some number at step **520**, say $10,000, one possible rule to be applied would be to ask if the input itself is formatted correctly to be usable. A numerical value would be mechanically correct, whereas some other entry, such as a text entry, would not fit the format expected for a vehicle value, and would be rejected. The rule would fail.

Many rules associated with the pre-method control point are merely means to validate entries and format. Rule **1** in block **550** is an example. If rule 1 fails, the

process would flow through to step **560**, and exit the method without saving the new value. However, other types of rules do more than merely check entries for mechanical condition. Specific business function rules, such as whether or not the entry is sensible, are also possible rule types.

In the preferred embodiment, another possibility is that the rule will call up data from the object such as year, make, model, and condition of the vehicle, and compare a calculated value against the value entered initially. Rule **2** in block **550** is an example of such a business type rule. If the value is sensible, the program proceeds on.

There are still other types of rules that have no direct correlation to the value itself, but are important to the business or process as a whole. A third type of rule (step **550**) may start an asynchronous audit process. This process may run independently of the other two rules being implemented. Examples of these might be to check the account for instances of fraud in the past. This rule may initiate a completely different sequence of events prescribed by other entities within the business itself.

A rule for auditing the method would probably be associated with a post-method context. The results of the post-method rule might point to changes needed in the rules or needed in implementation of the rules to ensure that the rules validate the method.

Another possible rule, which has very little to do with the value of the vehicle itself, but may be

important to the insurance company, would be to provide the policy holder making the claim with other information pertaining to other products available from the company. So, for instance, if a claim against a vehicle is being concluded and a check for the full value is being cut, a rule might cause particular literature on home, health, and other policies the insurance company offers to be inserted in the envelope with the check.

Figure 5 illustrates how methods in objects relate to their pre-method and post-method control points, and to the rules associated with them by showing a representative example. Each object has a number of methods. **Figure 5** illustrates one such method (step **600**). During the course of program execution, when the method is invoked (step **602**), execution encounters a pre-method control point (step **604**). This control point is shown in expanded form (step **606**).

The control point uses its selection algorithm (step **608**) to select the relevant set of rules (step **610**) based on the class of the object, the name of the method, and the fact that the control point is a pre-method control point. It then fires the rules one by one, passing each rule a reference to the object of which this method is a part, and the parameters that the method was passed when it was invoked.

Each rule calculates its result so that after firing the rules, the control point has a set of rule-firing results, which it combines into a single result using its combining algorithm (step **612**). The combining algorithm may differ from control point to control point depending

on the needs of the situation. Frequently, if the rules return boolean results, it will simply carry out a 'logical and' or 'logical or' of the rule-firing results. The combining algorithm decides, based on the combined result, whether to continue by executing the method body (step **614**), or to bypass method body execution and return (step **626**).

If the combining algorithm decides to continue with method body execution, it must also decide whether to make the combined result available to the method body. Frequently, if the combined result is a boolean value, it does not, but if the combined result is of some other data type, it does. If the combining algorithm decides that execution is to bypass the method body execution, it may optionally (and usually does) throw an exception so that the invoker of the method can detect that the method has not been executed due to a rule-based decision.

Assuming that the method body is to be executed, control flows into it at step **614**. After the method body is executed, the post-method control point is encountered (steps **616** and **618**). The post-method control point operates in much the same way that the pre-method control point does, including rule selection (step **620**), rule firing (step **622**), and results combining (step **624**). The major difference is that, because the method body has already been executed, the rule combination algorithm's options for dealing with the combined result are largely limited to throwing an exception.

Whether or not the method body is executed and whether or not an exception is thrown, in the end the method returns (step **624**) to its invoker.

Docket No. AT9-98-266

CLAIMS:

What is claimed is:

1. A computer implemented process for applying a set of rules, the process comprising:
   (a) placing a pre-method control before logic of a method and post-method control point after the logic of the method;
   (b) associating a set rules with each control point based on a class of object in which the method resides, name of the method, and type of control point, whether the pre-method control point or the post-method control point;
   (c) invoking the method, wherein encountering each control point during the execution of the method comprises:
      (i) determining if the encountered control point is active;
      (ii) on the basis of an active control point:
         1) selecting rules based on a set of rules associated with the active control point associated in step (b);
         2) running the selected rules;
         3) obtaining results from running the rules; and
         4) combining the results using a combining algorithm specified by the control point.

2. A computer implemented process for applying a set of rules comprising:

(a) defining an object;

(b) defining at least one method in the object;

(c) defining a control point just before logic of at least one method; and

(d) associating a set of rules with the control point.

3. In the process of claim 2, the step of defining a first control point further comprises:

(a1) decorating the object to dynamically insert a first control point such that the object acquires this new control point.

4. In the process of claim 2, the step of defining at least one control point further comprises:

(c1) adding the at least one control point through the technique of generating required code in the compiler or with a preprocessor.

5. In the process of claim 2, the step of defining at least one control point further comprises:

(c1) manually inserting the at least one control point and encoding the control point in the implementation of a hosting object.

6. In the process of claim 2, the step of defining at least one control point further comprises:

(c1) externalizing the at least one control point as a class and instantiating it at the at least one control point.

7. The process of claim 2 further comprises:

(e) defining a second control point just after the logic of each method; and

(f) associating a second set of rules with the second control point.

8. In the process of claim 7, wherein the rules in the second set of rules are associated to the second control point without considering the rules in the first set of rules associated with the at least one control point.

9. In the process of claim 7, wherein a set of rules is defined as having N number of rules, N being at least zero.

10. In the process of claim 2, the step of associating at least one control point further comprises:

(c1) defining, with a control point, at least one of a rule selecting algorithm and a rule-results combination algorithm.

11. The process of claim 2, further comprises:

(e) changing rules associated with the control point contained in the set of rules.

12. A computer implemented process for applying a set of rules, comprising:

(a) invoking a method in an object;

(b) encountering an active control point during the invocation of the method;

(c) selecting rules associated with the method of the object at the control point;

(d) invoking the rules; and

(e) combining results from invoking the rules.

13. The process of claim 12, wherein the rules perform a variety of actions conditioned by the fact that rules may be associated with particular, regularly occurring points in the object model.

14. The process of claim 12, wherein the rules perform at least one function which varies over time.

15. A process of claim 12, wherein a control point occurs just before logic of the method begins, just after the logic of the method completes, or at both just before logic of the method begins and just after the logic of the method completes.

16. A computer implemented process for applying a set of rules comprising:

(a) defining an object;

(b) defining at least one method in the object;

(c) defining at least one control point in the at least one method;

(d) defining rules to the at least one control point on basis the object's class name, method, name, and position of the at least one control point in the method.

17. In the process of claim 16, further comprising the step of activating at least one control point having associated rules.

18. The process of claim 16 further comprising:

(e) encountering a first control point;

(e) running the rules associated with the first control point; and

(f) affecting behavior of the object base on running the rules associated with the first control point.

19. In the process of claim 18, the step of affecting the behavior of the object further comprises:

(i) associating different rules to a control point.

20. In the process of claim 18, the step of affecting the behavior of the object further comprises:

(i) defining another control point.

21. In the process of claim 18, the step of modifying the object further comprises:

(i) associating rules to a second control point.

22. In the process of claim 16, further comprising a step of deactivating the at least one control point.

23. A computer implemented process for applying a set of rules, comprising

(a) defining an object;

(b) defining a method in the object;

(c) defining a first control point of the method;

(d) determining rules associated with the first control point;

(e) defining a second control point of the method; and

(f) determining rules associated with the second control point.

24. A computer implemented process as in claim 23 further comprising:

(g) separately selecting, running and combining the results of rules determined to be associated with either control point.

25. In the process of claim 23 wherein the first control point is a pre-method trigger point.

26. In the process of claim 23 wherein the second control point is a post-method trigger point.

27. A computer implemented process for defining an object comprising:
defining an object;
defining a method in the object by:
defining method logic;
placing the method logic in the method;
defining at least one control point; and
placing the at least one control point in the method wherein the method logic is continuous.

28. A computer implemented process for defining an object as in claim 27, wherein the step of placing the at least one control point further comprises placing the at least one control in the method before the method logic.

29. A computer implemented process for defining an object as in claim 27, wherein the step of placing the at least one control point further comprises placing the at least one control in the method after the method logic.

30. A computer implemented process for defining an object as in claim 27, wherein the at least one control comprises two control points and further comprising:
placing a first control in the method before the method logic; and
placing a second control in the method after the method logic.

31. A computer implemented process for defining an object as in claim 27, further comprises:
flagging the at least one control point on the basis of being active.

32. A computer implemented process for defining an object as in claim 27, wherein the step of defining the at least one control point further comprising:
defining a rule selection algorithm associated with the at least one control point.

33. A computer implemented process for defining an object as in claim 27, wherein the step of defining the at least one control point further comprising:
defining a rule result combination algorithm associated with the at least one control point.

34. A computer implemented process for defining an object as in claim 27, wherein the step of defining the at least one control point further comprises:
defining a rule selection algorithm for the at least one control point; and
defining a rule result combination algorithm for the at least one control point.

35. A computer implemented process for defining an object as in claim 27, further comprising:
associating at least one rule with the at least one control point.

36. A computer implemented process for defining a rule comprising:
creating the rule;
associating the rule with an object class;
associating the rule with a method within the object class; and
associating the rule with an occurrence of a control point within the method.

37. A computer implemented process for defining a rule as in claim 36 wherein the occurrence of the control point within the method being before method logic.

38. A computer implemented process for defining a rule as in claim 36 wherein the occurrence of control point within the method being after method logic.

39. A computer implemented process for defining a rule as in claim 36, further comprising:
associating the rule with another object class.

40. A computer implemented process for defining a rule as in claim 36, further comprising:
associating the rule with another method within the object class.

41. A computer implemented process for defining a rule as in claim 36, further comprising:
associating the rule with another control point within the method of the object class.

42. A computer implemented process for applying a set of rules, comprising:
selecting an object class;
selecting a method within the object class;
invoking the method;
processing rules associated with the method comprising:
encountering a control point associated with the method;
determining if the control point is active; and
finding at least one rule associated with an active control point.

43. A computer implemented process for applying a set of rules as in claim 42, wherein the step of finding at least one rule further comprises:

accessing a selecting algorithm associated with the active control point; and

selecting at least one rule using the selecting algorithm.

44. A computer implemented process for applying a set of rules as in claim 42, where in the step of processing rules further comprises:

running the at least one rule;

determining results from running the at least one rule;

accessing a combining algorithm associated with the control point; and

combining the results using the combining algorithm.

45. A computer implemented process for applying a set of rules, comprising:

selecting an object class;

selecting a method within the object class;

invoking the method;

processing rules comprising:

encountering a control point;

accessing a selecting algorithm associated with the control point; and

selecting at least one rule using the selecting algorithm.

46. A computer implemented process for applying a set of rules, comprising:

selecting an object class;

selecting a method within the object class;

invoking the method;

processing rules comprising:

encountering a control point;

finding at least one rule associated with the control point;

running the at least one rule;

determining results on the basis of running the at least one rule;

accessing a combining algorithm associated with the control point; and

combining the results using the combining algorithm.

47. A computer implemented process for applying a set of rules, comprising:

selecting an object class;

selecting a method within the object class;

invoking the method;

processing rules comprising:

encountering a first control point associated with the method;

determining if the first control point is active;

executing method logic of the method;

encountering a second control point associated with the method;

determining if the second control point is active;

finding a set of rules associated with one of the first control point and the second control point, wherein the set of rules contains not less than zero rules.

48. A computer implemented process for applying a set of rules, comprising:
selecting an object class;
selecting a method within the object class;
invoking the method;
processing rules comprising:
encountering a control point associated with the method;
finding at least one rule associated with the control point prior to executing method logic of the method;
running the at least one rule;
obtaining results on the basis of running the at least one rule; and
controlling the method on the basis of the results.

49. A computer implemented process for applying a set of rules as in claim 48, wherein the step of controlling the method comprises:
exiting the method.

50. A computer implemented process for applying a set of rules as in claim 48, wherein the step of controlling the method comprises:
executing method logic of the method.

51. A data processing system for defining an object comprising:
defining means for defining an object;
defining means for defining a method in the object by:
defining means for defining method logic;
placing means for placing the method logic in the method;
defining means for defining at least one control point; and
placing means for placing the at least one control point in the method wherein the method logic is continuous.

52. A data processing system for defining an object as in claim 51, wherein the step of placing the at least one control point further comprises placing means for placing the at least one control in the method before the method logic.

53. A data processing system for defining an object as in claim 51, wherein the step of placing the at least one control point further comprises placing means for placing the at least one control in the method after the method logic.

54. A data processing system for defining an object as in claim 51, wherein the at least one control comprises two control points and further comprising:
placing means for placing a first control in the method before the method logic; and
placing means for placing a second control in the method after the method logic.

55. A data processing system for defining an object as in claim 51, further comprises:
flagging means for flagging the at least one control point on the basis of being active.

56. A data processing system for defining an object as in claim 51, wherein the step of defining the at least one control point further comprising:
defining means for defining a rule selection algorithm associated with the at least one control point.

57. A data processing system for defining an object as in claim 51, wherein the step of defining the at least one control point further comprising:
defining means for defining a rule result combination algorithm associated with the at least one control point.

58. A data processing system for defining an object as in claim 51, wherein the step of defining the at least one control point further comprises:
defining means for defining a rule selection algorithm for the at least one control point; and
defining a rule result combination algorithm for the at least one control point.

59. A data processing system for defining an object as in claim 51, further comprising:
associating means for associating at least one rule with the at least one control point.

60. A data processing system for defining a rule comprising:
creating means for creating the rule;
associating means for associating the rule with an object class;
associating means for associating the rule with a method within the object class; and
associating means for associating the rule with an occurrence of a control point within the method.

61. A data processing system for defining a rule as in claim 60 wherein the occurrence of the control point within the method being before method logic.

62. A data processing system for defining a rule as in claim 60 wherein the occurrence of control point within the method being after method logic.

63. A data processing system for defining a rule as in claim 60, further comprising:
associating means for associating the rule with another object class.

64. A data processing system for defining a rule as in claim 60, further comprising:
associating means for associating the rule with another method within the object class.

65. A data processing system for defining a rule as in claim 60, further comprising:
associating means for associating the rule with another control point within the method of the object class.

66. A data processing system for applying a set of rules, comprising:
selecting means for selecting an object class;
selecting means for selecting a method within the object class;
invoking means for invoking the method;
processing means for processing rules associated with the method comprising:
encountering means for encountering a control point associated with the method;
determining means for determining if the control point is active; and
finding means for finding at least one rule associated with an active control point.

67. A data processing system for applying a set of rules as in claim 66, wherein the step of finding at least one rule further comprises:

accessing means for accessing a selecting algorithm associated with the active control point; and

selecting means for selecting at least one rule using the selecting algorithm.

68. A data processing system for applying a set of rules as in claim 66, where in the step of processing rules further comprises:

running means for running the at least one rule;

determining means for determining results from running the at least one rule;

accessing means for accessing a combining algorithm associated with the control point; and

combining means for combining the results using the combining algorithm.

69. A data processing system for applying a set of rules, comprising:

selecting means for selecting an object class;

selecting means for selecting a method within the object class;

invoking means for invoking the method;

processing means for processing rules comprising:

encountering means for encountering a control point;

accessing means for accessing a selecting algorithm associated with the control point; and

selecting means for selecting at least one rule using the selecting algorithm.

Docket No. AT9-98-266

70. A data processing system for applying a set of rules, comprising:

selecting means for selecting an object class;

selecting means for selecting a method within the object class;

invoking means for invoking the method;

processing means for processing rules comprising:

- encountering means for encountering a control point;
- finding means for finding at least one rule associated with the control point;
- running means for running the at least one rule;
- determining means for determining results on the basis of running the at least one rule;
- accessing a combining algorithm associated with the control point; and
- combining means for combining the results using the combining algorithm.

Docket No. AT9-98-266

71. A data processing system for applying a set of rules, comprising:

selecting means for selecting means for selecting an object class;

selecting means for selecting means for selecting a method within the object class;

invoking means for invoking the method;

processing means for processing rules comprising:

encountering means for encountering a first control point associated with the method;

determining means for determining if the first control point is active;

executing means for executing method logic of the method;

encountering means for encountering a second control point associated with the method;

determining means for determining if the second control point is active;

finding means for finding a set of rules associated with one of the first control point and the second control point, wherein the set of rules contains not less than zero rules.

72. A data processing system for applying a set of rules, comprising:
selecting means for selecting an object class;
selecting means for selecting a method within the object class;
invoking means for invoking the method;
processing means for processing rules comprising:
encountering means for encountering a control point associated with the method;
finding means for finding at least one rule associated with the control point prior to executing method logic of the method;
running the at least one rule;
obtaining means for obtaining results on the basis of running the at least one rule; and
controlling means for controlling the method on the basis of the results.

73. A data processing system for applying a set of rules as in claim 72, wherein the step of controlling the method comprises:
exiting means for exiting the method.

74. A data processing system for applying a set of rules as in claim 72, wherein the step of controlling the method comprises:
executing means for executing method logic of the method.

75. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for defining an object, the instruction comprising:
instructions for defining an object;
instructions for defining a method in the object by:
instructions for defining method logic;
instructions for placing the method logic in the method;
instructions for defining at least one control point; and
instructions for placing the at least one control point in the method wherein the method logic is continuous.

76. A computer program product for defining an object as in claim 75, wherein the step of placing the at least one control point further comprises placing the at least one control in the method before the method logic.

77. A computer program product for defining an object as in claim 75, wherein the step of placing the at least one control point further comprises placing the at least one control in the method after the method logic.

78. A computer program product for defining an object as in claim 75, wherein the at least one control comprises two control points and further comprising:
instructions for placing a first control in the method before the method logic; and
instructions for placing a second control in the method after the method logic.

79. A computer program product for defining an object as in claim 75, further comprises:
instructions for flagging the at least one control point on the basis of being active.

80. A computer program product for defining an object as in claim 75, wherein the step of defining the at least one control point further comprising:
instructions for defining a rule selection algorithm associated with the at least one control point.

81. A computer program product for defining an object as in claim 75, wherein the step of defining the at least one control point further comprising:
instructions for defining a rule result combination algorithm associated with the at least one control point.

82. A computer program product for defining an object as in claim 75, wherein the step of defining the at least one control point further comprises:
instructions for defining a rule selection algorithm for the at least one control point; and
instructions for defining a rule result combination algorithm for the at least one control point.

83. A computer program product for defining an object as in claim 75, further comprising:
instructions for associating at least one rule with the at least one control point.

84. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for defining a rule, the instruction comprising:
instructions for creating the rule;
instructions for associating the rule with an object class;
instructions for associating the rule with a method within the object class; and
instructions for associating the rule with an occurrence of a control point within the method.

85. A computer program product for defining a rule as in claim 84 wherein the occurrence of the control point within the method being before method logic.

86. A computer program product for defining a rule as in claim 84 wherein the occurrence of control point within the method being after method logic.

87. A computer program product for defining a rule as in claim 84, further comprising:
instructions for associating the rule with another object class.

88. A computer program product for defining a rule as in claim 84, further comprising:
instructions for associating the rule with another method within the object class.

89. A computer implemented process for defining a rule as in claim 84, further comprising:
instructions for associating the rule with another control point within the method of the object class.

90. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for applying a set of rules, the instruction comprising:
instructions for selecting an object class;
instructions for selecting a method within the object class;
instructions for invoking the method;
instructions for processing rules associated with the method comprising:
instructions for encountering a control point associated with the method;
instructions for determining if the control point is active; and
instructions for finding at least one rule associated with an active control point.

91. A computer program product for applying a set of rules as in claim 90, wherein the step of finding at least one rule further comprises:
instructions for accessing a selecting algorithm associated with the active control point; and
instructions for selecting at least one rule using the selecting algorithm.

92. A computer program product for applying a set of rules as in claim 90, where in the step of processing rules further comprises:

instructions for running the at least one rule;

instructions for determining results from running the at least one rule;

instructions for accessing a combining algorithm associated with the control point; and

instructions for combining the results using the combining algorithm.

93. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for applying a set of rules, the instruction comprising:

instructions for selecting an object class;

instructions for selecting a method within the object class;

instructions for invoking the method;

instructions for processing rules comprising:

instructions for encountering a control point;

instructions for accessing a selecting algorithm associated with the control point; and

instructions for selecting at least one rule using the selecting algorithm.

94. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for applying a set of rules, the instruction comprising:

instructions for selecting an object class;

instructions for selecting a method within the object class;

instructions for invoking the method;

instructions for processing rules comprising:

instructions for encountering a control point;

instructions for finding at least one rule associated with the control point;

instructions for running the at least one rule;

instructions for determining results on the basis of running the at least one rule;

instructions for accessing a combining algorithm associated with the control point; and

instructions for combining the results using the combining algorithm.

95. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for applying a set of rules, the instruction comprising:

instructions for selecting an object class;

instructions for selecting a method within the object class;

instructions for invoking the method;

instructions for processing rules comprising:

instructions for encountering a first control point associated with the method;

instructions for determining if the first control point is active;

instructions for executing method logic of the method;

instructions for encountering a second control point associated with the method;

instructions for determining if the second control point is active;

instructions for finding a set of rules associated with one of the first control point and the second control point, wherein the set of rules contains not less than zero rules.

96. A computer program product embodied on a computer readable medium containing instructions for a computer implemented process for applying a set of rules, the instruction comprising:
instructions for selecting an object class;
instructions for selecting a method within the object class;
instructions for invoking the method;
processing rules comprising:
instructions for encountering a control point associated with the method;
instructions for finding at least one rule associated with the control point prior to executing method logic of the method;
instructions for running the at least one rule;
instructions for obtaining results on the basis of running the at least one rule; and
instructions for controlling the method on the basis of the results.

97. A computer program product for applying a set of rules as in claim 96, wherein the step of controlling the method comprises:
instructions for exiting the method.

98. A computer program product for applying a set of rules as in claim 96, wherein the step of controlling the method comprises:
instructions for executing method logic of the method.

Docket No. AT9-98-266

ABSTRACT OF THE DISCLOSURE

METHOD AND APPARATUS FOR APPLYING BUSINESS RULES IN AN OBJECT MODEL DRIVEN CONTEXT

A means is provided for specifying, applying, and managing sets of temporary or permanent additions or modifications to the behavior of object-oriented programs without having to change the code of the program, by using externalized rules. The points at which the externalized rules may be applied is determined by the implementation object model, thus making their specification natural to the program developers who are familiar with the program's implementation object model. The concept of dynamic method-based trigger or control points is disclosed for identifying potential rule attachment points in objects and identifying the rules that are applicable to each dynamic control point.

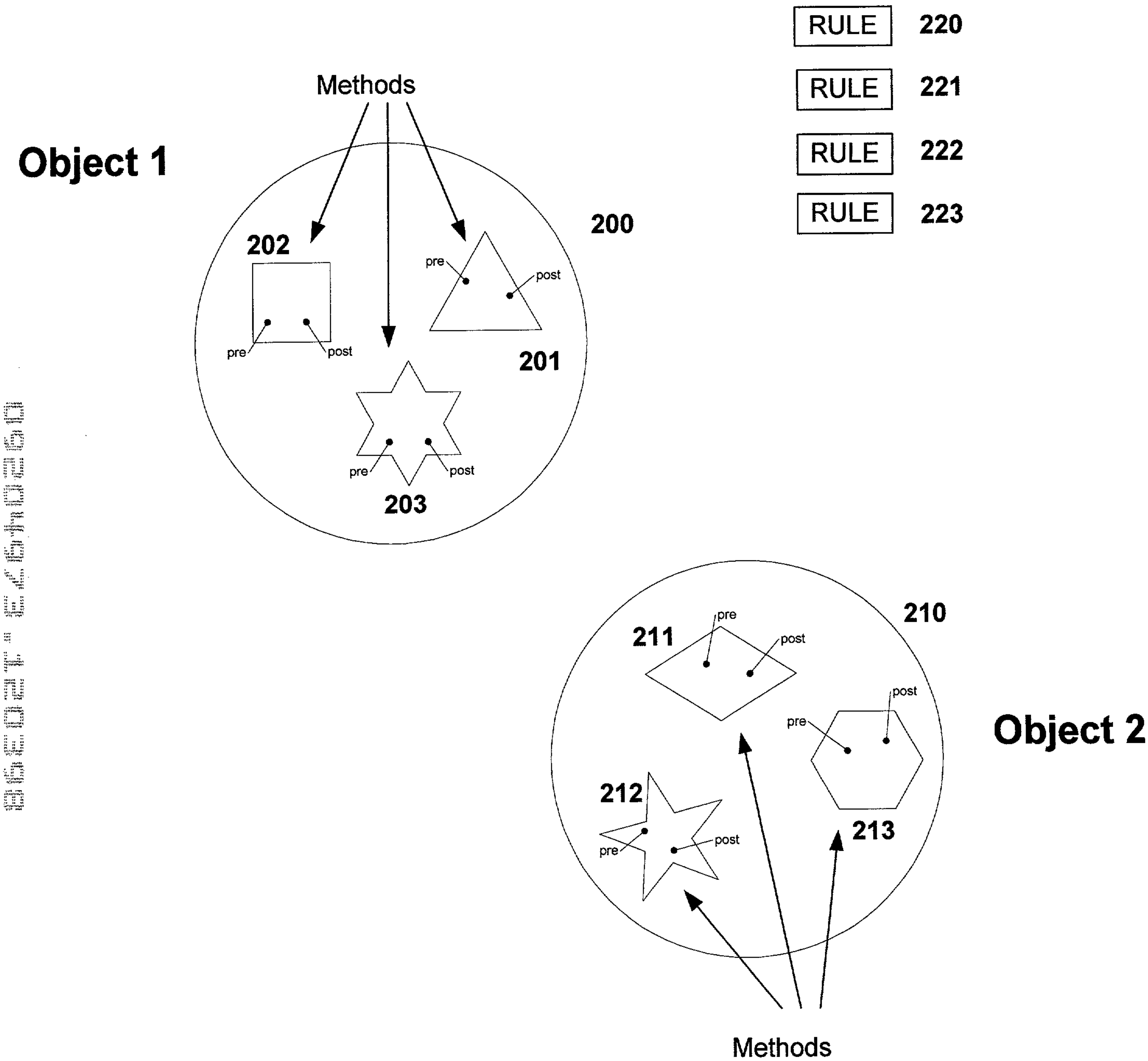

Figure 1

AT9-98-266

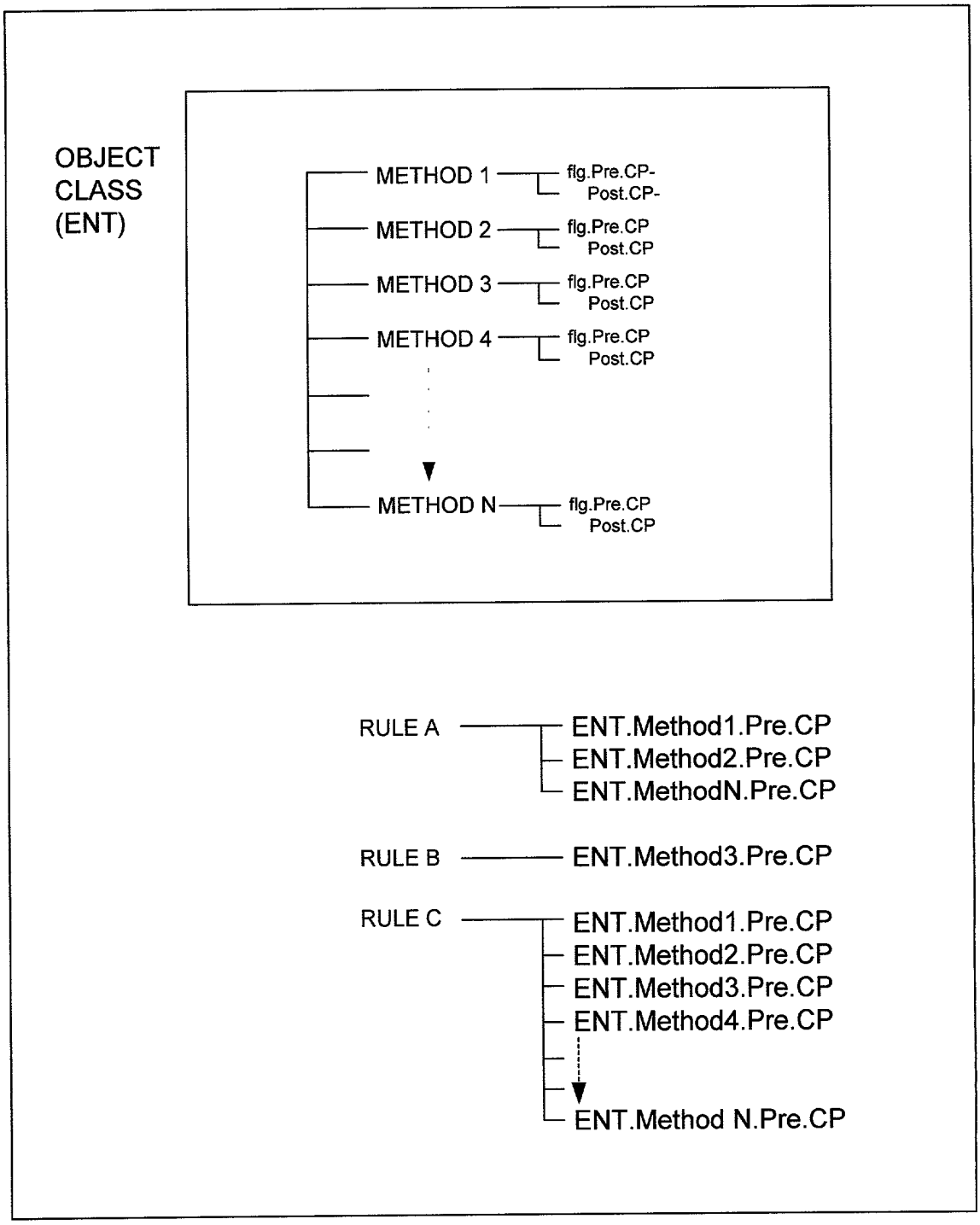

Figure 2

AT9-98-266

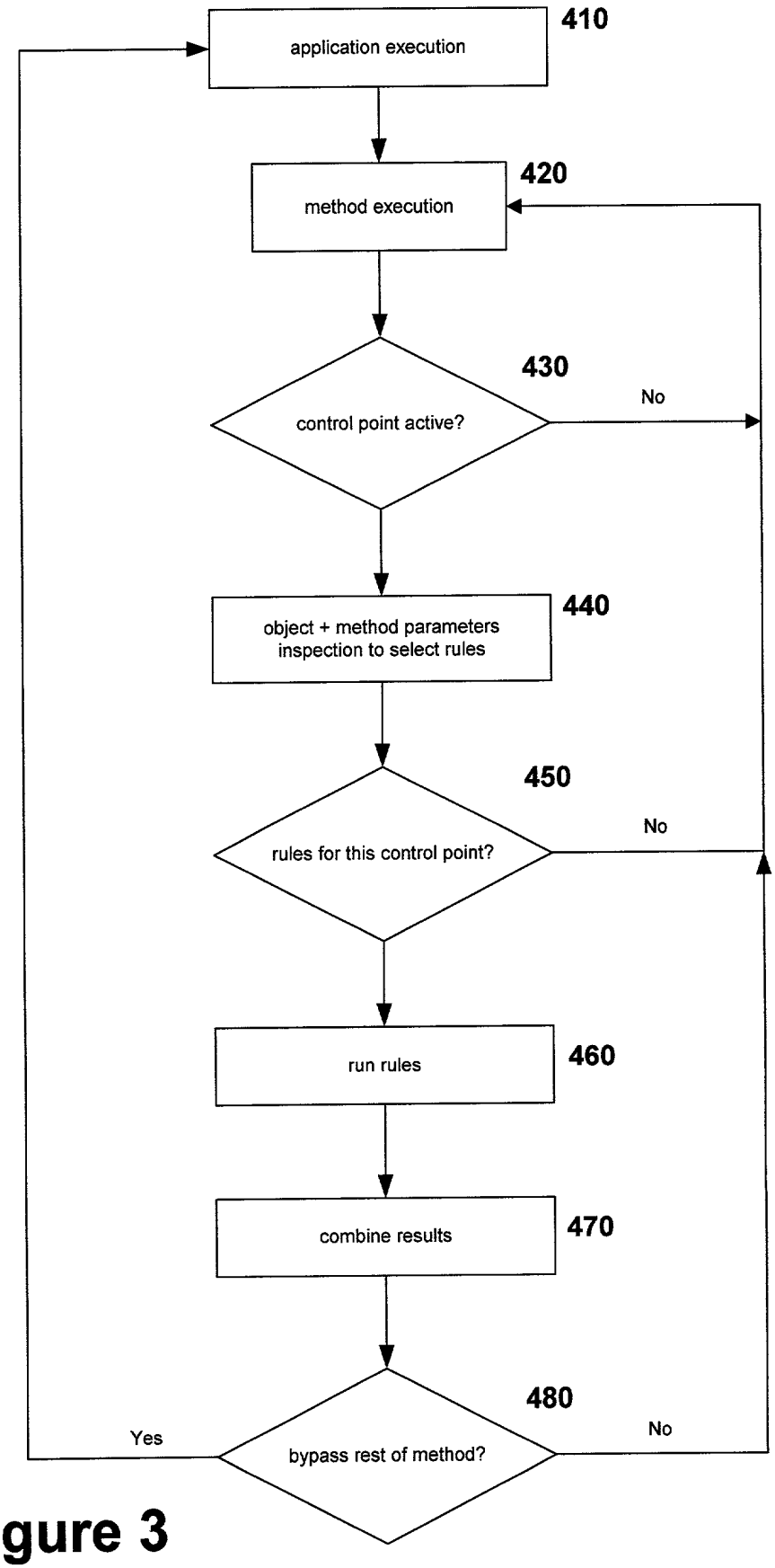

Figure 3

AT9-98-266

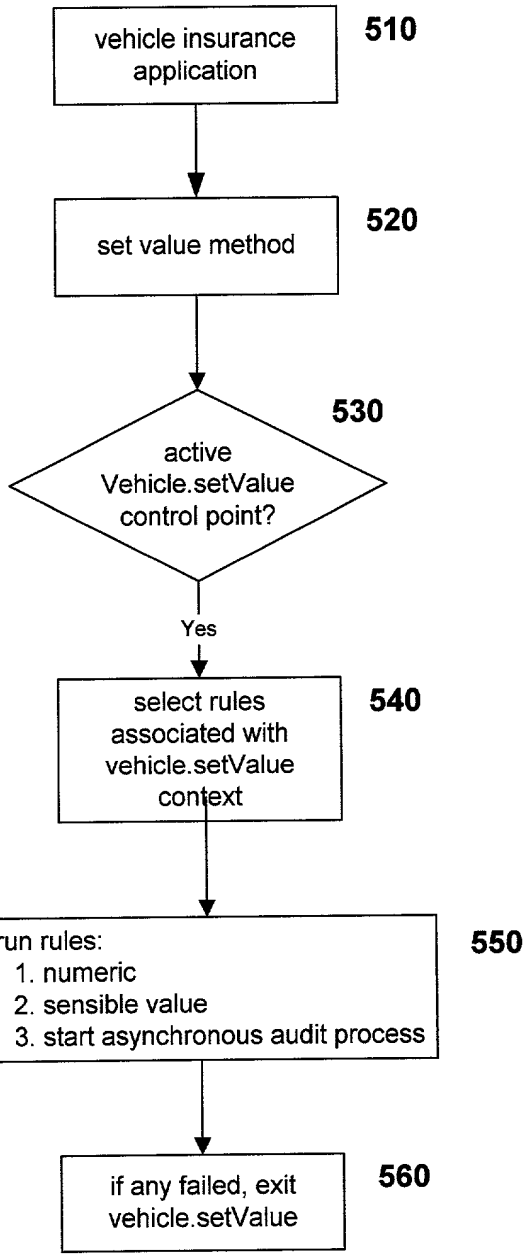

Figure 4

AT9-98-266

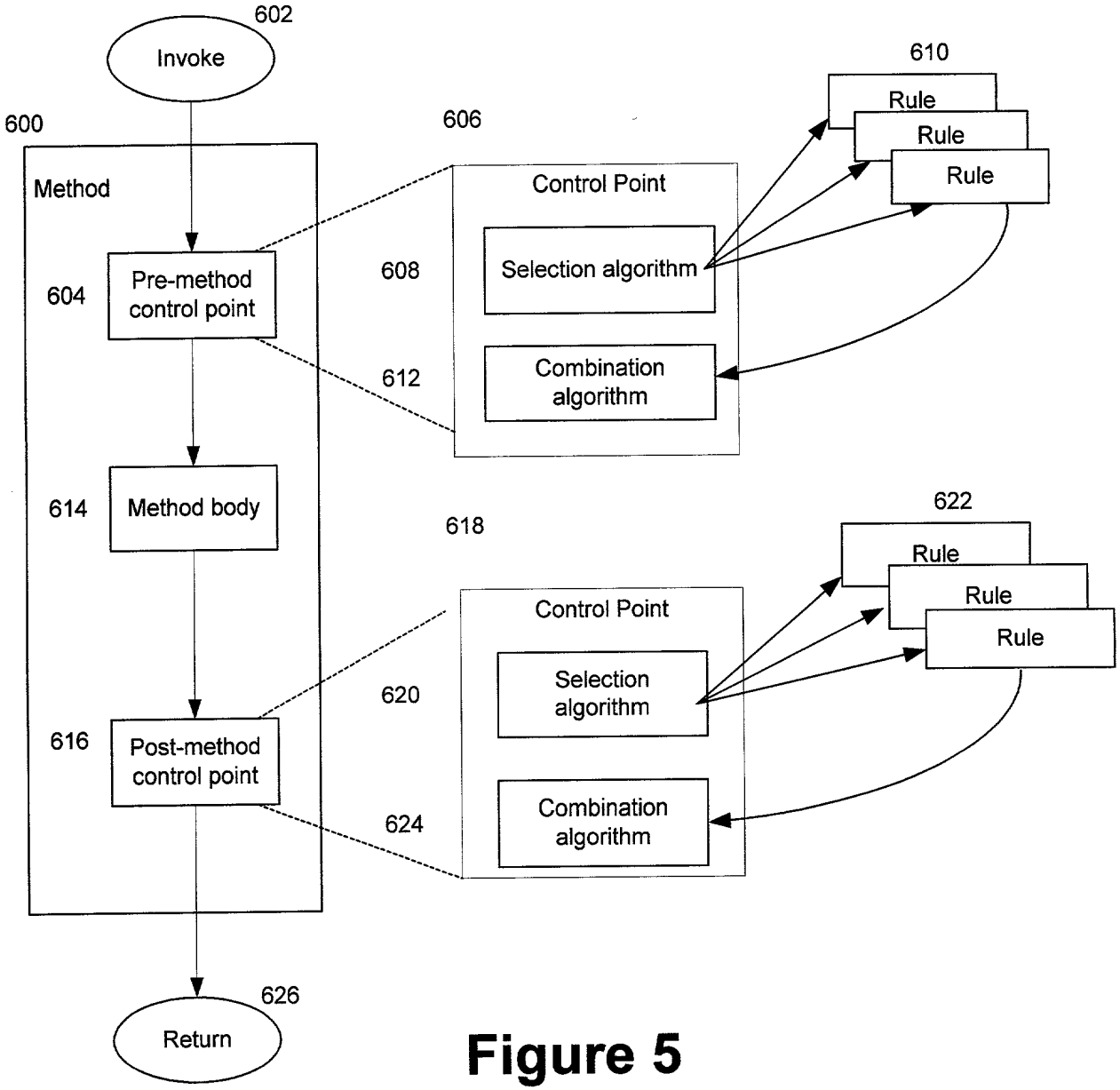

Figure 5

AT9-98-266

DOCKET NUMBER: AT9-98-266

DECLARATION AND POWER OF ATTORNEY FOR

PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Method and Apparatus for Applying Business Rules in an Object Model Driven Context

the specification of which (check one)

_X_ is attached hereto.

__ was filed on _____
as Application Serial No._____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, §1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, §119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):			Priority Claimed
_____ (Number)	_____ (Country)	_____ (Day/Month/Year)	___ Yes___ No

I hereby claim the benefit under Title 35, United States Code, §120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, §112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, §1.56 which occurred between the filing date of the prior application and the national or PCT international filing date of this application:

DOCKET NUMBER: AT9-98-266

(Application Serial #)	(Filing Date)	(Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; Thomas E. Tyson, Reg. No. 28,543; James H. Barksdale, Jr., Reg. No. 24,091; Casimer K. Salys, Reg. No. 28,900; Robert M. Carwell, Reg. No. 28,499; Douglas H. Lefeve, Reg. No. 26,193; Jeffrey S. LaBaw, Reg. No. 31,633; David A. Mims, Jr., Reg. 32,708; Volel Emile, Reg. No. 39,969; Richard A. Henkler, Reg. No. 39,220; and Anthony V. England, Reg. No. 35,129; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Duke W. Yee, Reg. No. 34,285; David W. Carstens, Reg. No. 34, 134; and Colin P. Cahoon, Reg. No. 38,836.

Send correspondence to: Duke W. Yee, Carstens, Yee & Cahoon, LLP, P.O. Box 802334, Dallas, Texas 75380 and direct all telephone calls to Duke W. Yee, (972) 362-2001

FULL NAME OF SOLE OR FIRST INVENTOR: David Lars Ehnebuske

INVENTORS SIGNATURE: David Lars Ehnebuske DATE: December 2, 1998

RESIDENCE: 105 Hideaway Cove
Georgetown, Texas 78628

CITIZENSHIP: USA

POST OFFICE ADDRESS: Same

FULL NAME OF SECOND INVENTOR: Barbara Jane Alspach McKee

INVENTORS SIGNATURE: Barbara Jane Alspach McKee DATE: December 2, 1998

RESIDENCE: 2203 Cypress Point East
Austin, Texas 78746

CITIZENSHIP: USA

POST OFFICE ADDRESS: Same

DOCKET NUMBER: AT9-98-266

FULL NAME OF THIRD INVENTOR: Isabelle Marie Catherine Rouvellou

INVENTORS SIGNATURE:____________________ DATE:________________

RESIDENCE: 225 W. 106th St Apt #8F
New York City, New York 10025

CITIZENSHIP: France

POST OFFICE ADDRESS: Same

DOCKET NUMBER: AT9-98-266

DECLARATION AND POWER OF ATTORNEY FOR

PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

Method and Apparatus for Applying Business Rules in an Object Model Driven Context

the specification of which (check one)

_X_ is attached hereto.

___ was filed on _____
as Application Serial No._____
and was amended on _____
(if applicable)

I hereby state that I have reviewed and understand the contents of the above identified specification, including the claims, as amended by any amendment referred to above.

I acknowledge the duty to disclose information which is material to the patentability of this application in accordance with Title 37, Code of Federal Regulations, '1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, '119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):			Priority Claimed
_____ (Number)	_____ (Country)	_____ (Day/Month/Year)	___ Yes___ No

I hereby claim the benefit under Title 35, United States Code, '120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, '112, I acknowledge the duty to disclose information material to the patentability of this application as defined in Title 37, Code of Federal Regulations, '1.56 which occurred between the filing date of the

DOCKET NUMBER: AT9-98-266

prior application and the national or PCT international filing date of this application:

(Application Serial #)	(Filing Date)	(Status)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: As a named inventor, I hereby appoint the following attorneys and/or agents to prosecute this application and transact all business in the Patent and Trademark Office connected therewith.

John W. Henderson, Jr., Reg. No. 26,907; Thomas E. Tyson, Reg. No. 28,543; James H. Barksdale, Jr., Reg. No. 24,091; Casimer K. Salys, Reg. No. 28,900; Robert M. Carwell, Reg. No. 28,499; Douglas H. Lefeve, Reg. No. 26,193; Jeffrey S. LaBaw, Reg. No. 31,633; David A. Mims, Jr., Reg. 32,708; Volel Emile, Reg. No. 39,969; Richard A. Henkler, Reg. No. 39,220; and Anthony V. England, Reg. No. 35,129; Leslie A. Van Leeuwen, Reg. No. 42,196; Christopher A. Hughes, Reg. No. 26,914; Edward A. Pennington, Reg. No. 32,588; John E. Hoel, Reg. No. 26,279; Joseph C. Redmond, Jr., Reg. No. 18,753; Marilyn S. Dawkins, Reg. No. 31,140; Duke W. Yee, Reg. No. 34,285; David W. Carstens, Reg. No. 34, 134; and Colin P. Cahoon, Reg. No. 38,836.

Send correspondence to: Duke W. Yee, Carstens, Yee & Cahoon, LLP, P.O. Box 802334, Dallas, Texas 75380 and direct all telephone calls to Duke W. Yee, (972) 362-2001

FULL NAME OF SOLE OR FIRST INVENTOR: David Lars Ehnebuske

INVENTORS SIGNATURE:_____ DATE:_____

RESIDENCE: 105 Hideaway Cove
Georgetown, Texas 78628

CITIZENSHIP: USA

POST OFFICE ADDRESS: Same

FULL NAME OF SECOND INVENTOR: Barbara Jane Alspach McKee

INVENTORS SIGNATURE:_____ DATE:_____

RESIDENCE: 2203 Cypress Point East

DOCKET NUMBER: AT9-98-266

Austin, Texas 78746

CITIZENSHIP: USA

POST OFFICE ADDRESS: Same

FULL NAME OF THIRD INVENTOR: Isabelle Marie Catherine Rouvellou

INVENTORS SIGNATURE: Isabelle Marie Catherine Rouvellou DATE: 12/02/1998

RESIDENCE: 225 W. 106th St Apt #8F
New York City, New York 10025

CITIZENSHIP: France

POST OFFICE ADDRESS: Same